STM Library API

Error codes returned by different functions: (Constants defined in STM.h)

Constant Name	Value	Description
SUCCESS	0	indicates success
FAILURE	-1	indicates failure
MEMORY_INSUFFICIENT	-2	indicates that memory is insufficient for new data objects creation
INVALID_TRANSACTION	-3	indicates that protocol specific validation failed for given transaction
TOB_ABSENT	-4	indicates that required data object does not exist in shared memory
SIZE_MISMATCH	-5	indicates that sizes of existing and required data objects do not match
TOB_ID_CLASH	-6	indicates that the id with which we want to create new data object already exists in shared memory

Some other constants defined in STM.h:

Constant Name	Value	Description
INITIAL_DATAITEMS	10	Used by default initializer to create data objects initially
DEFAULT_DATA_SIZE	4	indicates the size of default data objects created
HIGH_FREQ	1000 µs	Used by heavy applications to run garbage collection at high frequency
MEDIUM_FREQ	50000 µs	Sets the garbage collector to medium frequency
LOW_FREQ	100000 µs	Sets the garbage collector to low frequency

Selecting Protocol:

Syntax to initialize any protocol:

STM* <library_variable_name> = new <protocol_name>

Protocols available:

- SGT
- BTO
- MVTO

For instance, below is the syntax to initialize SGT protocol :

```
STM* lib = new SGT;
```

For protocols with Garbage collection, the following constructor is invoked :

STM* <library_variable_name> = new <protocol_name> (Garbage collection frequency)

For instance,

```
STM* lib = new SGT(HIGH_FREQ);
```

Note:

- *HIGH_FREQ, MEDIUM_FREQ and LOW_FREQ, as the names indicate, are the required frequencies of garbage collection defined as constants in STM.h file*
- *Among the above protocols, Garbage collection is applicable to SGT and MVTO.*

Data structures reference:

Struct	**Fields**	**Description**
Common_tob	Long long ID Void* value Int size	Indicates the name of the data object Indicates the value Indicates the size

trans_state	int TID;
	map<int,common_tOB*> local_buffer
	
	int read_flag;
		Transaction ID
		Local buffer to serve duplicate read and write requests
		Flag to check if any read operations were performed by the transaction

Note: *trans_state objects should not be tampered by the user.*

Function signatures:

S.No.	Parameters
1	trans_state* begin();
2	int initialize();
3	int initialize(vector<int> sizes);
4	int initialize(vector<int> initial_tobs,vector<int> sizes);
5	int read(trans_state* trans,common_tOB* tb);
6	int write(trans_state* trans,common_tOB* tb);
7	int try_commit(trans_state* trans,long long& error_tOB);
8	void try_abort(trans_state* trans);
9	int create_new(int& ID);
10	int create_new(int size, int& ID);
11	int create_new(long long ID,int size);
12	void* garbage_collector(void *lib);
13	void gc();
14	void end_library();

Detailed Function Descriptions:

S.No.	Function Name	Parameters	Return value	Description
1	begin	No parameters	trans_state*	creates a new trans_state object. Its pointer is returned to the user which is used as argument for later operations like "read" , "write" , "try_commit" and “try_abort”

S.No.	Function Name	Parameters	Return value	Description
2	initialize	No parameters	Int	Default memory initialization. Number of data items created and their default sizes are defined as constants in STM.h as mentioned above. IDs are sequential numbers starting from 0. Return value is the error_code. *Note: initialize function should be invoked only once. Else, new IDs created can not be determined. In protocols that require garbage collection , this method sets the gc_active variable to true and invokes garbage_collector thread which runs until gc_active variable is set to false.
3	initialize	Vector<int> sizes	Int	Same as above but user passes the sizes of data objects to be created through the sizes parameter. Number of data objects to be created is implicit from size of input.
4	initialize	1. Vector<int> initial_tobs 2.Vector<int> sizes	Int	Same as above but user also passes IDs of data objects to be created through the first parameter initial_tobs
5	read	1. trans_state* trans 2.common_tOB* tb	Int	Return value indicates the success/failure using error_codes defined above. Input parameter trans* indicates which transaction is performing read operation. Parameter tb indicates which data object is to be read. If read is successful, value field of tb is filled before return.
6	write	1. trans_state* trans 2.common_tOB* tb	Int	Return value indicates the success/failure using error_codes defined above. Input parameter trans* indicates which transaction is performing write operation. Parameter tb indicates which data object is to be written and the its value. In case of success, this value is written to actual shared memory.

S.No.	Function Name	Parameters	Return value	Description
7	try_commit	1.trans_state* trans 2.long long& error_tOB	Int	Return value indicates the success/failure using error_codes defined above. Input parameter trans indicates which transaction user is trying to commit. Reference parameter error_tOB is used by try_commit to inform the user about which data object is problematic in case of failure. In case of success, the required transaction commits.
8	try_abort	trans_state* trans	Void	Aborts the transaction indicated by input parameter trans
9	create_new	int& ID	Int	Creates new data object with default size and returns the ID of newly created object through input reference parameter, ID. The return value indicates error which can be found out using error codes defined above.
10	create_new	1.Int size 2. int& ID	Int	Same as above but here user passes the size of data object to be created using input parameter size.
11	create_new	1.Long long ID 2. int size	Int	Same as above but user passes ID of data object to be created using first input parameter ID.
12	garbage_collector	1.void*	void	User passes library object cast as void*. This thread invokes gc method periodically as per the frequency set in STM constructor until gc_active variable is set to false.
13	gc	No input parameters	void	Invoked by garbage collector thread periodically. It runs one cycle of garbage collection.
14	end_library	No parameters	void	In protocols using garbage collection, this method is used to set gc_active variable to false and thereby end the garbage collection thread.